IC - Project 1

Miguel Ferreira
João Fernandes
João Martins

18 de novembro de 2021

Conteúdo

1 Introduction

Relatório do primeiro projeto de Informação e Codificação onde iremos manipular texto, imagens e áudio. Iremos usar **AudioFile.h** para manipular ficheiros de áudio e **OpenCV** para as imagens. Iremos começar pela **Parte B** pois a **Parte A** é apenas assistir e a tutoriais e familiarizarmos-nos com a C++.

2 Parte B

2.1 Exercício 2

Q: Implement a program to copy a text file, character by character. Both file names should be passed as command line arguments to the program.

A: Requer dois argumentos dados pelo *user* (um para com o nome do ficheiro que quer copiar e outro com o nome do ficheiro onde ira ser inserida a copia)
Enquanto há informação no ficheiro vai copiando.
Acaba quando o ficheiro para copia acaba

2.2 Exercicio 3

Q: Implement a program to copy an audio file in wav format, sample by sample. Both file names should be passed as command line arguments to the program.

A: Usando o ficheiro fornecido **AudioFile.h** conseguimos facilmente dar *load* do ficheiro de áudio usando a função **.load(FilePath)** sendo que o FilePath é dado como argumento.
Para guardar o ficheiro usamos a função **.save(FileName, AudioFileFormat::Wave)** sendo o *FileName* dado como argumento e usamos o formate *Wave* para nos devolver um ficheiro *.wav*.

2.3 Exercício 4

Q: Implement a program to copy an image, pixel by pixel. Both file names should be passed as command line arguments to the program.

A: Para este exercício recorremos ao **OpenCV**. Para ler uma imagem usamos a função **imread(FilePath** sendo que o *FilePath* é passado como argumento.
Para criar uma cópia do ficheiro usamos a função **imwrite(FileName, FileToCopy** sendo que o *FileName* é dado como argumento e o *FileToCopy* é a variável usada para guardar os valores do aberto.

3 Parte C

3.1 Exercício 5

Q: Implement a program to calculate the histogram of the letters that exist in a text file and the corresponding entropy.

A: Para este programa são passados como argumentos o ficheiro para analisar e o ficheiro onde irá ser imprimido o histograma.
O programa consiste num **map<char, int>** que é um mapa que guarda os caracteres e o numero das suas ocorrências. Se o carater já existir no mapa é adicionado 1 ao valor das ocorrências, se não o carater é inserido no mapa com o valor 1 associado. Usamos como ficheiro para analisar *"Os Lusíadas"* que é passado como argumento

3.2 Exercício 6

Q: Implement a program to calculate the histogram of the audio samples in a sound file for both stereo and mono channels in a .wav sound file and the corresponding entropy.

A: O programa começa por construir dois histogramas, um para os canais stereo e outro para a média dos canais *stereo*. Após a construção dos histogramas, estes são usados para o calculo das entropias, *stereo* e mono respectivamente.
A construção do histograma *stereo* resulta do mapeamento das *samples* de ambos os canais e do numero da suas ocorrências. O histograma *mono* é construído de forma semelhante mas os valores das *samples* são convertidas para o seu valor absoluto e o numero de ocorrências uma média. O calculo da entropia é feito usando a fórmula: $-\sum_{i=1}^{N} prob(a_i) * \log_2 prob(a_i)$ sendo a_i o valor de uma *sample* e $prob(a_i) = NrOcorrencias_{a_i}/NTotalOcorrencias$.

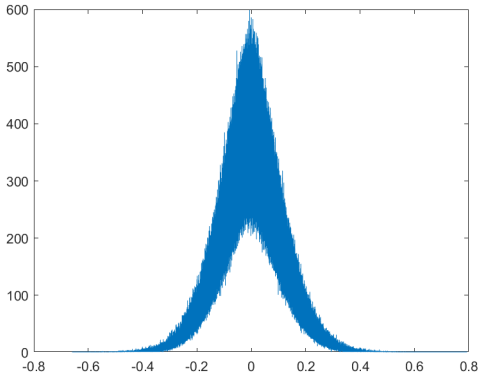


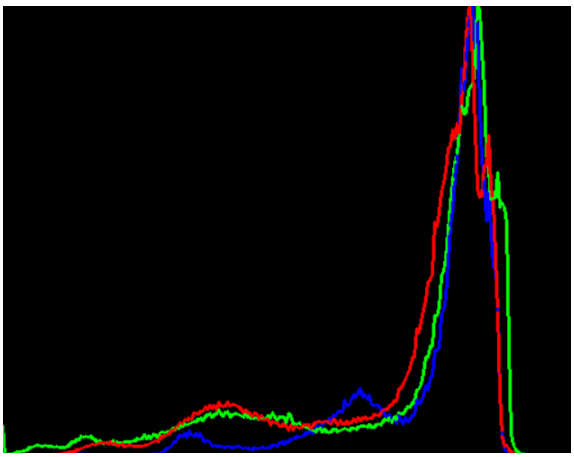
Figura 1: Histogram of all channels of *sample01.wav*

3.3 Exercício 7

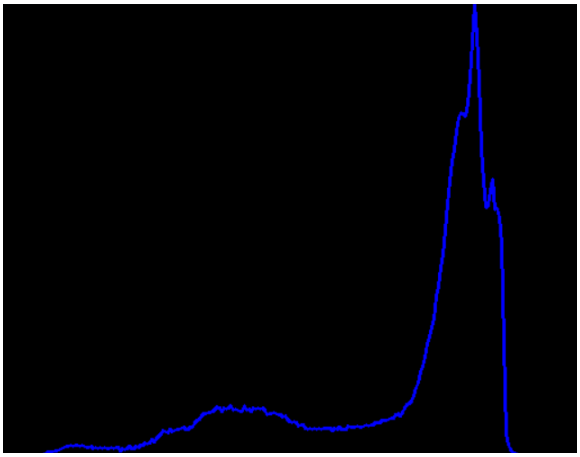
Q: Following the same suggestions of the previous exercise regarding the visualization, implement a program to calculate the histogram of an image file and the corresponding entropy.

A: Este programa calcula o histograma **RGB** e **GrayScale** de uma imagem submetida pelo utilizador. Começa por separar os valores de cor e de cinzentos, esses valores depois são processados, recorrendo a funções do opencv, dando um histograma contendo os valores de R, G, e B e outro com os valores de cinzentos.

Figura 2: Used Image



(a) RGB



(b) Gray Scale

Figura 3: Histograms

4 Parte D

4.1 Exercicio 8

Q: Implement a program to reduce the number of bits used to represent each audio sample.

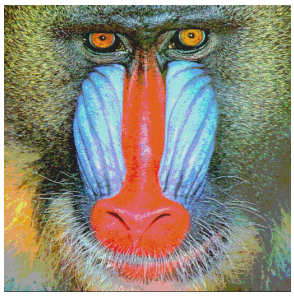
A: O programa implementado neste exercício reduz os bits necessários para representar uma *sample* dividindo essa *sample* por um múltiplo de 2 dado pela formula $division_factor = 2^{NrBits}$, com $NrBits$ o numero de bits a reduzir.

4.2 Exercício 9

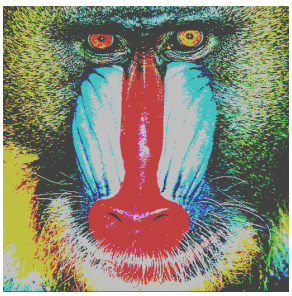
Q: Implement a program to reduce the number of bits used to represent each pixel of an image

A: Para este exercício podemos usar vários métodos para reduzir o numero de bits por cada imagem.
O primeiro que usamos é a divisão de cada pixel por fator de divisão constante. O programa leva como entrada o nome da imagem a ser processada e o factor de divisão.

(a) Original



(b) Fator de divisão 64



(c) Fator de divisão 128

Figura 4: Transformation

4.3 Exercício 10

Q: Implement a program that prints the signal-to-noise ratio (SNR) of a certain audio file in relation to the original file, as well as the maximum per sample absolute error.

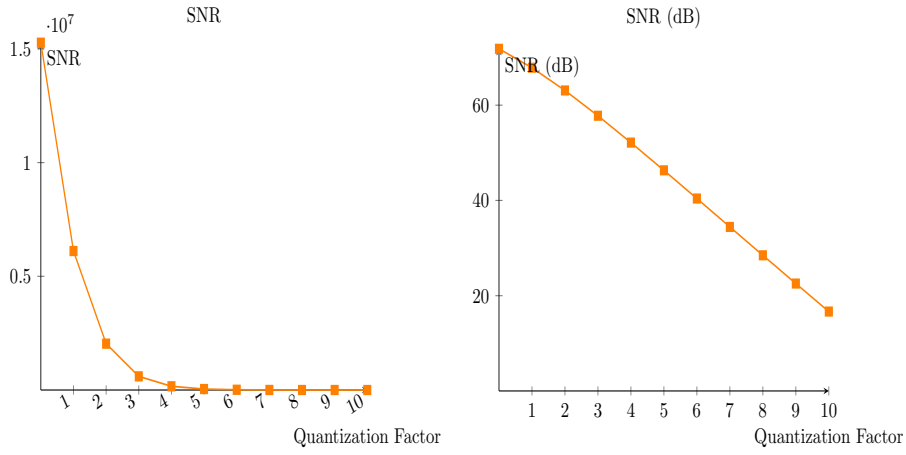
A: Implementação de um programa que recebe como argumentos o ficheiro de referencia, o ficheiro com o audio quantizado e u fator de quantização (numero de bits poupados). Durante a sua execução percorre as *samples* do audio quantizado e obtem um valor quantizado para comparação ao multiplicar o valor da *sample* com $2^{factor_quantizacao}$,

$SNR = \sum_{n=1}^{N} sinal(n) / \sum_{n=1}^{N} erro(n)$, sendo sinal(n) o valor da n^{th} *sample* e erro(n) a diferença entre o valor da n^{th} *sample* de referencia e o valor quantizado da n^{th} *sample* quantizada.

No Anexo 1 podem ser encontrados valores de SNR resultantes da quantização do ficheiro *sample01.wav*.

Quantization Factor	SNR	SNR (dB)
0	15271456.41	71.84
1	6112241.62	67.86
2	2038761.54	63.09
3	599808.35	57.78
4	163678.09	52.14
5	42826.93	46.32
6	10969.05	40.40
7	2782.79	34.44
8	704.97	28.48
9	179.73	22.55
10	46.42	16.67

Tabela 1: Resultados de SNR para diferentes fatores de quantização para a sample01.wav



4.4 Exercício 11

Q: Implement a program that prints the signal-to-noise ratio (SNR) of a certain image file in relation to the original file, as well as the maximum per pixel absolute error.

A: Este programa permite-nos calcular o PSNR entre duas figuras (uma original e outra alterada) recebendo-as como argumento. Para obter o PSNR usamos a formula, $PSNR = 10 \log \frac{A^2}{e^2}$ sendo $e = \frac{1}{N_R N_C} \sum_{i=1}^{N_R} \sum_{j=1}^{N_C} [f(i,j) - \tilde{f}(i,j)]^2$

Quantization Factor	SNR
20	32.8558
40	16.6925
60	16.489
80	16.1333
100	18.9295
120	15.2857
140	16.7724
160	15.4753
180	8.38903
200	10.525

Tabela 2: Resultados de SNR para diferentes fatores de quantização para a babbon.ppm